Package 'imola'

November 17, 2021

Type Package

Title CSS Layouts (Grid and Flexbox) Implementation for R/Shiny

Version 0.3.2

Description Allows easy creation of CSS layouts (grid and flexbox) directly from R without added CSS.

License MIT + file LICENSE

URL https://github.com/pedrocoutinhosilva/imola

Encoding UTF-8

Imports shiny, htmltools, magrittr, stringi, glue, yaml

Suggests testthat

RoxygenNote 7.1.2

NeedsCompilation no

Author Pedro Silva [aut, cre]

Maintainer Pedro Silva <pedrocoutinhosilva@gmail.com>

Repository CRAN

Date/Publication 2021-11-17 20:50:08 UTC

R topics documented:

`activeBreakpoints` *Current active media breakpoints. By default the default_system setting values are used, but the list can be customized by using the registerBreakpoint() and unregisterBreakpoint() functions.*

Description

Current active media breakpoints. By default the default_system setting values are used, but the list can be customized by using the registerBreakpoint() and unregisterBreakpoint() functions.

Usage

```
activeBreakpoints()
```

Value

A named list of media breakpoints options.

`applyTemplate` *Merges a set of attributes with a given template. To avoid redundanct attributes being added to the final list, a list of default values (based of the specific panel creation callback formals) is used to validate the need of the argument value in the final list.*

Description

Merges a set of attributes with a given template. To avoid redundanct attributes being added to the final list, a list of default values (based of the specific panel creation callback formals) is used to validate the need of the argument value in the final list.

Usage

```
applyTemplate(attributes, template, defaults, type)
```

Arguments

attributes	The manually given attribute values that will take priority during the merge.
template	The name of the template to merge.
defaults	The default values of the grid callback.
type	The type of css grid of the template.

Value

A named list of css attributes that can be used to generate a html element style rules of the given type.

---

flexPage *Create a page a with CSS flexbox layout*

Description

Create a page a with CSS flexbox layout

Usage

```
flexPage(..., title = NULL, fill_page = TRUE, dependency = bootstrapLib())
```

Arguments

...	Elements to include within the page
title	The browser window title (defaults to the host URL of the page)
fill_page	Flag to tell the page if it should adjust the page to adjust and fill the browser window size
dependency	The set of web dependencies. This value can be a htmlDependency, for example the shiny bootstrap one (the default) or a tagList with diferent dependencies

Value

A UI definition that can be passed to the [shinyUI] function.

Note

See https://css-tricks.com/snippets/css/a-guide-to-flexbox/ for additional details on using css flexbox

See Also

[flexPanel()]

Examples

```
if (interactive()) {
library(imola)
flexPage(
  title = "A flex page",
  div(class = "area-1"),
  div(class = "area-2"),
  div(class = "area-3")
)
}
```

`flexPanel` *Create a panel with a CSS flexbox layout*

Description

Create a panel with a CSS flexbox layout

Usage

```
flexPanel(
  ...,
  template = NULL,
  direction = "row",
  wrap = "nowrap",
  justify_content = "flex-start",
  align_items = "stretch",
  align_content = "flex-start",
  gap = 0,
  flex = c(1),
  grow = NULL,
  shrink = NULL,
  basis = NULL,
  breakpoint_system = activeBreakpoints(),
  id = generateID()
)
```

Arguments

`...`	Elements to include within the panel
`template`	The name of the template to use as a base for the grid. See listTemplates() and registerTemplate() for more information.

`direction` Direction of the flow of elements in the panel. Accepts a valid css 'flex-direction' value (row | row-reverse | column | column-reverse) By default the 'row' value is used. Supports named list for breakpoints.

`wrap` Should elements be allowed to wrap into multiple lines. Accepts a valid css 'flex-wrap' value (nowrap | wrap | wrap-reverse). By default the value 'wrap' is used. Supports named list for breakpoints.

`justify_content` Defines the alignment along the main axis. Accepts a valid css 'justify-content' value (flex-start | flex-end | center | space-between | space-around | space-evenly | start | end | left | right). By default the value 'flex-start' is used. Supports named list for breakpoints.

`align_items` Defines the default behavior for how flex items are laid out along the cross axis on the current line. Accepts a valid css 'align-items' value (stretch | flex-start | flex-end | center | baseline | first baseline | last baseline | start | end | self-start | self-end). By default the value 'stretch' is used. Supports named list for breakpoints.

`align_content` Aligns a flex container's lines within when there is extra space in the cross-axis. Accepts a valid css align-content' value (flex-start | flex-end | center | space-between | space-around | space-evenly | stretch | start | end | baseline | first baseline | last baseline). By default the value 'flex-start' is used. Supports named list for breakpoints.

`gap` Defines the space between elements in the panel. Defaults to 0. Supports named list for breakpoints.

`flex` A vector of valid css 'flex' values. Defines how elements in the panel can grow and shrink. Each entry of the vector will afect the nth child of the panel, meaning that it can be at maximum the lenght of the elements in the panel. If smaller the vector will be repeated until the pattern affects all elements in the panel. NA can also be used as a entry of the vector to skip adding a rule to specific elements. By default c(1) is used, meaning all elements can grow and shrink as required, at the same rate. See notes for more information. Supports named list for breakpoints.

`grow` A vector of valid css 'flex-grow' values. Defines the rate of how elements can grow. Entries will overwrite the 'flex' values, and can be used make more targeted rules. Each entry of the vector will afect the nth child of the panel, meaning that it can be at maximum the lenght of the elements in the panel. If smaller the vector will be repeated until the pattern affects all elements in the panel. NA can also be used as a entry of the vector to skip adding a rule to specific elements. By default c() is used, meaning values from the flex argument will be used. See notes for more information. Supports named list for breakpoints.

`shrink` A vector of valid css 'flex-shrink' values. Defines the rate of how elements can shrink. Entries will overwrite the 'flex' values, and can be used make more targeted rules. Each entry of the vector will afect the nth child of the panel, meaning that it can be at maximum the lenght of the elements in the panel. If smaller the vector will be repeated until the pattern affects all elements in the panel. NA can also be used as a entry of the vector to skip adding a rule to specific elements. By default c() is used, meaning values from the flex argument will be used. See notes for more information. Supports named list for breakpoints.

`basis` A vector of valid css 'flex-basis' values. Defines the base size of elements. Entries will overwrite the 'flex' values, and can be used make more targeted rules. Each entry of the vector will afect the nth child of the panel, meaning that it can be at maximum the lenght of the elements in the panel. If smaller the vector will be repeated until the pattern affects all elements in the panel. NA can also be used as a entry of the vector to skip adding a rule to specific elements. By default c() is used, meaning values from the flex argument will be used. See notes for more information. Supports named list for breakpoints.

`breakpoint_system` Optional Media breakpoints to use. Will default to the current active breakpoint system.

`id` The panel id. A randomly generated one is used by default.

Details

Behaves similar to normal HTML div tags, but simplifies the way css flexbox can be used from shiny.

Value

An HTML tagList.

Note

When creating responsive layouts based on media rules, for most arguments a named list can be passed instead of a single value. The names in the list can be any of the registered breakpoints available in activeBreakpoints(). Breakpoints can also be modified with the registerBreakpoint() and unregisterBreakpoint() functions. Arguments that allow this are: direction | wrap | justify_content | align_items | align_content | gap | flex | grow | shrink | basis.

See https://css-tricks.com/snippets/css/a-guide-to-flexbox/ for additional details on using css flexbox

See Also

[flexPage()]

Examples

```
if (interactive()) {
library(imola)
flexPanel(
  div("example content"),
  div("example content"),
  div("example content")
)
}
```

`generateCSSPropertyStyles` *Generates the requires css statements for a specific css property. It will iterated over all breakpoints in the given value and return all statements for all breakpoints in a vector format.*

Description

Generates the requires css statements for a specific css property. It will iterated over all breakpoints in the given value and return all statements for all breakpoints in a vector format.

Usage

```
generateCSSPropertyStyles(value, property, id, breakpoint_system)
```

Arguments

`value`	Normalized attribute value list to generate css from.
`property`	The type of grid to generate css for.
`id`	The id of the parent wrapper element.
`breakpoint_system`	Media breakpoints to use.

Value

A vector of valid css strings.

`generateFlexChildrenCSS` *Generates all required css for a set of children attributes, for a flex wrapper.*

Description

Generates all required css for a set of children attributes, for a flex wrapper.

Usage

```
generateFlexChildrenCSS(attributes, id, number_children, breakpoint_system)
```

Arguments

attributes	Normalized attribute list to generate css from.
id	The id of the parent wrapper element.
number_children	Number of child elements in the wrapper. Required to generate rules for flex elements.
breakpoint_system	Media breakpoints to use.

Value

A string with all placeholders replaced.

---

generateFlexCSS *Generates all required css for a set of attributes, for a flex wrapper.*

Description

Generates all required css for a set of attributes, for a flex wrapper.

Usage

```
generateFlexCSS(attributes, id, number_children, breakpoint_system)
```

Arguments

attributes	Normalized attribute list to generate css from.
id	The id of the parent wrapper element.
number_children	Number of child elements in the wrapper. Required to generate rules for flex elements.
breakpoint_system	Media breakpoints to use.

Value

A string with all placeholders replaced.

`generateGridAreaCSS` *Generates the requires css statements for a specific set of grid areas. This includes the css required to position each child element into each of the named grid area.*

Description

Generates the requires css statements for a specific set of grid areas. This includes the css required to position each child element into each of the named grid area.

Usage

```
generateGridAreaCSS(areas, id)
```

Arguments

`areas`	Unique names of areas for which css rules must be generated.
`id`	The id of the parent wrapper element.

Value

A vector of valid css strings.

`generateGridCSS` *Generates all required css for a set of attributes, for a grid wrapper.*

Description

Generates all required css for a set of attributes, for a grid wrapper.

Usage

```
generateGridCSS(attributes, id, unique_areas, breakpoint_system)
```

Arguments

`attributes`	Normalized attribute list to generate css from.
`id`	The id of the parent wrapper element.
`unique_areas`	Unique names of areas that the grid contains.
`breakpoint_system`	Media breakpoints to use.

Value

A string with all placeholders replaced.

`generateID` *Generates a HTML valid ID. HTML IDs should follow specific standards: - The ID must start with a letter (a-z or A-Z). - All subsequent characters can be letters, numbers (0-9), hyphens (-), underscores (_), colons (:), and periods (.). - Each ID must be unique within the document.*

Description

Generates a HTML valid ID. HTML IDs should follow specific standards: - The ID must start with a letter (a-z or A-Z). - All subsequent characters can be letters, numbers (0-9), hyphens (-), underscores (_), colons (:), and periods (.). - Each ID must be unique within the document.

Usage

```
generateID()
```

Value

A valid CSS id.

`gridPage` *Create a page a with CSS grid layout*

Description

Create a page a with CSS grid layout

Usage

```
gridPage(..., title = NULL, fill_page = TRUE, dependency = bootstrapLib())
```

Arguments

`...`	Elements to include within the page
`title`	The browser window title (defaults to the host URL of the page).
`fill_page`	Flag to tell the page if it should adjust the page to adjust and fill the browser window size.
`dependency`	The set of web dependencies. This value can be a htmlDependency, for example the shiny bootstrap one (the default) or a tagList with diferent dependencies

Value

A UI definition that can be passed to the [shinyUI] function.

Note

See https://css-tricks.com/snippets/css/complete-guide-grid/ for additional details on using css grids

See Also

[gridPanel()]

Other grid functions: `gridPanel()`

Examples

```
if (interactive()) {
library(imola)
gridPage(
  title = "A grid page",
  areas = c("area-1 area-1", "area-2 area-3"),
  div(class = "area-1"),
  div(class = "area-2"),
  div(class = "area-3")
)
}
```

`gridPanel` *Create a panel with a CSS grid layout*

Description

Create a panel with a CSS grid layout

Usage

```
gridPanel(
  ...,
  template = NULL,
  areas = NULL,
  rows = NULL,
  columns = NULL,
  gap = NULL,
  align_items = "stretch",
  justify_items = "stretch",
  auto_fill = TRUE,
  breakpoint_system = activeBreakpoints(),
  id = generateID()
)
```

Arguments

...	Elements to include within the panel. If areas are used, named named arguments using the grid area name will be added to that grid area. If no named arguments or areas are used, non attribute elements will be added to existing grid cells based on their order.
template	The name of the template to use as a base for the grid. See listTemplates() and registerTemplate() for more information.
areas	A list of vectors with area names, or a vector or strings representing each row of the grid. Each element should contain the names, per row, of each area of the grid. Expected values follow the convension for the grid-template-areas css attribute. for example c("area-1 area-1", "area-2 area-3") and list(c("area-1", "area-1"), c(area-2", "area-3")) are both valid representations of a 2x2 grid with 3 named areas. If using breakpoints, a named list of valid values where each name is a valid breakpoint can be used as well. For example list(default = c("area-1", "area-2"), xs = list(c("area-1"), c("area-2"))) would be a valid value for breakpoints
rows	A string of css valid sizes separated by a space. or a vector of sizes. For example both "1fr 2fr" or c("1fr", "2fr") are valid representations of the same 2 rows grid sizes. Follows the convension for the grid-template-rows css attribute. If not provided the existing space will be split equally acording to the areas defined in areas. Supports named list for breakpoints.
columns	A string of css valid sizes separated by a space. or a vector of sizes. For example both "1fr 2fr" or c("1fr", "2fr") are valid representations of the same 2 columns grid sizes. Follows the convension for the grid-template-columns css attribute. If not provided the existing space will be split equally acording to the areas defined in areas. Supports named list for breakpoints.
gap	The space (in a valid css size) between each grid cell. Supports named list for breakpoints.
align_items	The cell behavior according to the align-items css property. Defaults to stretch. Supports named list for breakpoints.
justify_items	The cell behavior according to the justify-items css property. Defaults to stretch. Supports named list for breakpoints.
auto_fill	Should the panel stretch to fit its parent size (TRUE), or should its size be based on its children element sizes (FALSE).
breakpoint_system	Optional Media breakpoints to use. Will default to the current active breakpoint system.
id	The html id of the panel container.

Details

Behaves similar to normal HTML div tags, but simplifies the way css grid can be used via the arguments area, rows and columns. Only areas is required, when not used rows and columns will simply split the existing space equaly between each row / column. Internally the function creates the styles for the grid positions by generating the rules for positioning the children via their classes. To position a child element simply make sure that it includes a class with the same name as the named area.

Value

An HTML tagList.

Note

When creating responsive layouts based on media rules, for arguments a named list can be passed instead of a single value. The names in the list can be any of the registered breakpoints available in activeBreakpoints(). Breakpoints can also be modified with the registerBreakpoint() and unregisterBreakpoint() functions. Arguments that allow this are: areas | rows | columns | gap.

See https://css-tricks.com/snippets/css/complete-guide-grid/ for additional details on using css grids

See Also

[gridPage()]

Other grid functions: `gridPage()`

Examples

```
if (interactive()) {
library(imola)
gridPanel(
  areas = c("area-1 area-1", "area-2 area-3"),
  rows = "1fr 2fr",
  columns = "2fr 100px",
  div(class = "area-1"),
  div(class = "area-2"),
  div(class = "area-3")
)
}
```

`listTemplates` *Lists all available grid and flex templates. If type is given, returns only templates for the given grid type. Templates are collections of arguments that can be grouped and stored for later usage via the "template" argument of panel and page functions.*

Description

Lists all available grid and flex templates. If type is given, returns only templates for the given grid type. Templates are collections of arguments that can be grouped and stored for later usage via the "template" argument of panel and page functions.

Usage

```
listTemplates(type = NULL)
```

Arguments

type	Optional argument for what type of css templates to return. value must be either "grid" or "flex". If no type is given, all templates of all types are returned,

Value

A named list of css templates and specific values.

---

mediaRuleTemplate *Generates a valid glue::glue string template for a css media query. Used internally to generate a breakpoint specific wrapper.*

Description

Generates a valid glue::glue string template for a css media query. Used internally to generate a breakpoint specific wrapper.

Usage

```
mediaRuleTemplate(options)
```

Arguments

options	The options for the required template. if no valid values are given, a non media query template is created instead.

Value

A valid glue::glue template string to be processed later.

---

normalizeAttribute *Converts non named list attributes into a named list. Does nothing if the attribute is already a list in the correct format.*

Description

Converts non named list attributes into a named list. Does nothing if the attribute is already a list in the correct format.

Usage

```
normalizeAttribute(attribute, simplify = TRUE)
```

Arguments

attribute	The value to process
simplify	Boolean flag if the attribute should be simplified into single strings.

Value

A named list.

---

normalizeAttributes *Applies the normalizeAttribute() function to a full attribute list.*

Description

Applies the normalizeAttribute() function to a full attribute list.

Usage

```
normalizeAttributes(attributes)
```

Arguments

attributes	The values to process

Value

A named list.

---

processContent *Adds a css class to any HTML elements from the content that are named and which name is in the areas vector for names. This allows content to be assigned to the grid areas via named argument while still allowing other generic HTML tag attributes to be used.*

Description

Adds a css class to any HTML elements from the content that are named and which name is in the areas vector for names. This allows content to be assigned to the grid areas via named argument while still allowing other generic HTML tag attributes to be used.

Usage

```
processContent(content, areas)
```

Arguments

content	A (named) list of HTML elements.
areas	The names from content that should have a class added.

Value

A list of HTML elements.

---

readSettingsFile *Reads the content of a YAML settings file from the package directory.*

Description

Reads the content of a YAML settings file from the package directory.

Usage

```
readSettingsFile(file)
```

Arguments

file	The file name to read. Settings files are stored in the package installation directory and include diferent settings and options.

Value

A list object containing the content of the settings YAML file

---

registerBreakpoint *Adds a new breakpoint entry to the currelty active media breakpoints.*

Description

Adds a new breakpoint entry to the currelty active media breakpoints.

Usage

```
registerBreakpoint(name, min = NULL, max = NULL)
```

Arguments

name	The name of the entry to remove
min	The minimum screen width (in pixels) when the rule is active
max	The maximum screen width (in pixels) when the rule is active

Value

No return value, called for side effects

registerTemplate *Registers a new css template for future use. Depending on the given type, the template will then be available to be passed as an argument to a panel or page function of that specific type. Templates are collections of arguments that can be grouped and stored for later usage via the "template" argument of panel and page functions.*

Description

Registers a new css template for future use. Depending on the given type, the template will then be available to be passed as an argument to a panel or page function of that specific type. Templates are collections of arguments that can be grouped and stored for later usage via the "template" argument of panel and page functions.

Usage

```
registerTemplate(
  type,
  name,
  ...,
  breakpoint_system = activeBreakpoints(),
  export = NULL
)
```

Arguments

type	The type of css grid for which the template can be used
name	A unique name for the template. If a template with the same type and name exists, it will simply be overwritten.
...	Collection of valid arguments that can be passed to a panel of the given type (see gridPanel() and FlexPanel() for all options)
breakpoint_system	Optional breakpoint system to use in the template. by default it will simply use the current active system but a built in or custom system can also be passed. You ca find built in breakpoint systems under getOption("imola.breakpoints")
export	A file name to export the template to. Allows exporting templates as a yaml file for future usage.

Value

No return value, called for side effects

`setBreakpointSystem` *Sets the current active media breakpoints. By default the default_system setting values are used, but the list can be customized by using the registerBreakpoint() and unregisterBreakpoint() functions.*

Description

Sets the current active media breakpoints. By default the default_system setting values are used, but the list can be customized by using the registerBreakpoint() and unregisterBreakpoint() functions.

Usage

```
setBreakpointSystem(system)
```

Arguments

`system`	The name of the media breakpoints system to use. Existing systems can be found under getOption("imola.breakpoints")

Value

A named list of media breakpoints options.

`stringCSSRule` *Applies a CSS statement template stored in the package settings. These templates use the htmlTemplate format, meaning placeholders are marked using the placeholder convention. Each placeholder value should be passed as a named argument to the function using the placeholder value as a name. Used primarily as a shorthand to stringTemplate() for stored templates.*

Description

Applies a CSS statement template stored in the package settings. These templates use the htmlTemplate format, meaning placeholders are marked using the placeholder convention. Each placeholder value should be passed as a named argument to the function using the placeholder value as a name. Used primarily as a shorthand to stringTemplate() for stored templates.

Usage

```
stringCSSRule(template, ...)
```

Arguments

`template`	The template name to use. Available templates are saved in options, under getOption("imola.settings")$string_templates.
`...`	Named arguments to use in the template string. All placeholders in the template must have a corresponding named argument.

Value

A valid CSS string.

`stringTemplate` *Processes a string template in the htmlTemplate format into a valid string with no placeholders. The string must use the htmlTemplate format, meaning placeholders are marked using the placeholder convention.*

Description

Processes a string template in the htmlTemplate format into a valid string with no placeholders. The string must use the htmlTemplate format, meaning placeholders are marked using the placeholder convention.

Usage

```
stringTemplate(string, ...)
```

Arguments

`string`	The string template. Uses the same format as the htmlTemplate() function from shiny. placeholders in the template should use the placeholder format.
`...`	Named arguments to use in the template string. All placeholders in the template must have a corresponding named argument.

Value

A string with all placeholders replaced.

`unregisterBreakpoint` *Allows removing an entry from the current activeBreakpoints.*

Description

Allows removing an entry from the current activeBreakpoints.

Usage

```
unregisterBreakpoint(name)
```

Arguments

`name`	The name of the entry to remove

Value

No return value, called for side effects

`unregisterTemplate` *Deletes an existing css template from the available list of templates for the given grid type. Templates are collections of arguments that can be grouped and stored for later usage via the "template" argument of panel and page functions.*

Description

Deletes an existing css template from the available list of templates for the given grid type. Templates are collections of arguments that can be grouped and stored for later usage via the "template" argument of panel and page functions.

Usage

```
unregisterTemplate(type, name)
```

Arguments

`type`	The type of css grid of the template to remove.
`name`	The name of the tempalte to remove.

Value

No return value, called for side effects

valueToCSS	*Converts a R List or vector object into a valid css string. Used primarily to convert normalized attribute values into css values during processing.*

Description

Converts a R List or vector object into a valid css string. Used primarily to convert normalized attribute values into css values during processing.

Usage

```
valueToCSS(value, property)
```

Arguments

value	The list or vector with the values to be converted into css
property	The target css property for which the value will be used.

Value

string containing a valid css value.

Index